MESA Intro and White Dwarf Lab

Ryan Connolly
Michigan State University

September 9, 2014

Introduction

This is a brief introductory tutorial on getting started with MESA from scratch, learning some of the basics, and getting a feel for working with accreting white dwarfs. It's designed for beginners who have little to no experience using MESA, but should *not* be treated as comprehensive. That is, I'll focus on white dwarf models and novae specifically, but you should feel free to explore other aspects of MESA as well!

This tutorial is written for MESA version 6794 or later, and assumes you setup MESA using the MESA SDK by Rich Townsend (found here). If you have never used MESA before, Section 1 of this tutorial will guide you through installation. If you already have a functional MESA installation that fits the aforementioned requirements, you can skip to Section 2.

As a warning, my own MESA experience is almost exclusively on Mac with a mix of terminal usage and browsing with the "Finder". If you're using Linux or remotely accessing another machine you'll probably require more terminal practice/experience than I'll discuss here (apologies!).

Please send all questions and comments to rpconnol@gmail.com!

Contents

1 Installation and Setup

If you have MESA version 6794 installed already, you can skip to Section 2.

The MESA Sourceforge page already has a walkthrough for installing and compiling MESA, which is now easier than ever.

http://mesa.sourceforge.net/prereqs.html

This page should guide you through installation for your Mac or Linux machine. You can optionally forgo the preliminary stuff and skip down to Section 1.5, "Install the prerequisites (MESA SDK)" (at the above link).

Once you've successfully installed MESA, it would be very useful to continue on and do as much of the Getting Started tutorial as you can.

http://mesa.sourceforge.net/starting.html

The above guide covers the absolute basics of getting started, creating a fresh working directory, familiarizing yourself with running MESA and interpreting the output, etc. I'll go over many of the same things in the later sections of this tutorial as well, but it never hurts to have more practice!

If you run into any issues with the installation process, or have trouble with any of your first runs, email me and I'll do my best to help troubleshoot.

2 Example Test Suite Case

3 Building a Baseline Model

4 Suggested Further Reading

Below is a few additional useful tutorials and labs that I've compiled from the 2014 MESA Summer School and other sources.

Frank Timmes' `pgstar` plotting tutorial

Download the zip file from this link:

http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/timmes

Once extracted, you can find the lecture slides in the `presentations_pdf_files` folder. Ignore all the individual pdf's and just use `mesa_ss_2014.pdf`. The presentation/lab is nicely structured and starts off holding your hand with extremely detailed steps, but then leaves more and more up to the reader as the lab progresses. Perfect for learning `pgstar` as well as just general practice for experienced users. Keep in mind that when working with MESA in the future for your own research you can make one "master" `pgstar` inlist that has your favorite plots and easily copy it for multiple projects! After finishing the lab, it might be worthwhile investing some time creating your own plots with information relevant to your research using what you've just learned from the lab as a reference.

Kevin Moore's general MESA tutorial and `run_star_extras` lab

The lecture is here:

http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/moore

and the tutorial/lab itself (also linked in the lecture) is located here:

http://users.soe.ucsc.edu/~klmoore/index.html

Part 1 of the lab is a great intro even for relatively new MESA users. For someone with moderate MESA experience (months+), Parts 1 and 2 are both great practice as well as insightful. You may or may not be familiar with most of the things the first two parts cover, but the succinct and clean tutorial helped me wrap my head around a few things even after using MESA for over a year. Part 3 about adding physics hooks is more advanced and probably only necessary if you require it for your research.

Josiah Schwab's `run_star_extras` tutorial

Similar to Kevin Moore's tutorial above. Either of the following links should work.

https://github.com/jschwab/rse_tutorial

or

http://yoshiyahu.org/mesa2013.html

Again, just more good practice, and perhaps you'll get something out of it that you wouldn't in Kevin's, or vice versa.

Bill Wolf's `MesaScript` and `PyMesaReader`

An impromptu intro Bill threw together for and evening session at the 2014 MESA Summer School.

http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/evening-session-talks

(first entry, Evening_1_MesaScript_PyMesaReader.pdf)

`PyMesaReader` is probably more relevant. It's one of several tools floating around that helps get history and profile data output from MESA into Python easily and (relatively) intuitively. Everyone has different ways of doing this, and Python is flexible enough that each work pretty much as well as the next, but I know when I was new it could often be a pain and distraction, so it's nice to have tools/instructions on the ready.

`MesaScript` is a Ruby-based tool designed to let you use typical programming commands/variables/logical statements to make MESA inlists. For example, if you want to make a series of runs but want each star to have slightly different masses, `MesaScript` will allow you to output a bunch of organized inlists (e.g. "`inlist_1M`, `inlist_2M`, `inlist_4M`, `inlist_8M`, etc") with all of the same matching controls and commands except for the initial mass of the star. There are great examples in the PDF. On smaller scales this is something most of us have done at one point or another in our projects. While it's probably easier to do manually for a couple cases, if you wanted to run a large grid of simulations with dozens of inlists, `MesaScript` would help substantially in keeping things organized and (importantly!) ensuring all of your inlists match! When working with multiple inlists, it's easy to adjust controls to test something in only one place and forget that you did so down the road. Even if you aren't familiar with Ruby but have other Python/programming experience, it should be intuitive enough and the documentation and examples are good (and likely to be updated occasionally in the future as well).